CS280

Fall 2016

Midterm Exam

Name: \_\_\_\_\_\_\_\_\_\_KEY\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Section: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**SECTION I**

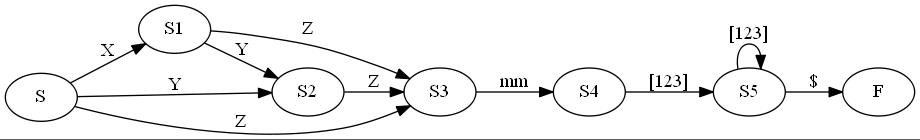
**Select the best choice and write the letter for your choice in the box provided. (20)**

|  |  |  |
| --- | --- | --- |
| 1. | C | Dereferencing an uninitialized pointer in a C++ program:   1. is impossible because it is checked at runtime 2. always evaluates to 0 3. may result in an invalid memory reference 4. is permitted only under Microsoft Windows 5. none of the above |
| 2. | D | After executing the code fragment below, what is the result?  int var = 8;  int\* vr = var;  var += 3;  cout << \*vr;   1. a runtime error occurs because vr is not initialized 2. the value 8 is printed 3. the value 10 is printed 4. a compile time error occurs at the initialization of vr 5. a null pointer exception is thrown |
| 3. | B | The type name enclosed in < > when using one of the STL classes means   1. nothing; it is basically a comment 2. the STL class is type checked 3. performance is at its highest possible level 4. finding an item in a collection object is faster 5. none of the above |
| 4. | B | An enumerated type in C++   1. is an integer or a string valued type 2. lists all possible values of an instance of the type 3. cannot be used with the == operator 4. B and C 5. None of the above |
| 5. | D | A nondeterministic automata   1. Runs more slowly than a deterministic automata 2. Cannot perform metacharacter matches 3. Can only be used in a text editor 4. May give different results for the same input string 5. Does not exist |
| 6. | C | A parser resolves conflicts in an expression grammar by using   1. Precedence rules 2. Associativity rules 3. A and B 4. A or B but not both 5. Neither A nor B |
| 7. | B | The “if”, “while” and “class” in C++ are examples of:   1. Identifiers 2. Keywords 3. Variables 4. Control Structures 5. None of the above |
| 8. | D | A static variable declared inside of a function means:   1. There is one memory binding for the variable per function call 2. The memory for the variable is on the heap, not the stack 3. The program cannot take the address of the variable 4. There is one memory binding for the variable, established before run time 5. The variable cannot be passed to a function |
| 9. | D | When an operator is overloaded   1. The built in operators are disabled 2. The programmer specifies which operator to use by calling a function directly 3. The precedence of the operator changes 4. The compiler uses types of operands to determine the operation 5. None of the above |
| 10. | B | Strong typing means   1. The compiler finds type errors and corrects them 2. The compiler finds type errors and stops the compile 3. The programmer is lazy 4. Only primitive types can be used 5. Type casting is disabled |

**SECTION II**

**Answer the following questions (30)**

1. Draw a DFA representing the regular expression given below. Represent the end of the string with the symbol $  
     
    X?Y?Zmm[123]+



1. Given the grammar below:  
     
   E -> E + T | E – T | T  
   T -> T \* P | T / P | P  
   P -> id | ( E )  
     
   Rewrite the grammar to add the operators < and >. These operators have the same precedence; that precedence is higher than + but lower than \*

E -> E + C | E – C | C  
C -> C > T | C < T | T  
T -> T \* P | T / P | P  
P -> id | ( E )

1. Given the grammar below, provide a leftmost derivation for the given input string:  
     
   TOKENS: ***INT*** for integers, ***VAR*** for single letter, +. \*, (, )  
     
   GRAMMAR:

E ::= T { + E }

T ::= P { \* T }

P ::= ***INT*** | ***VAR*** | ( E )  
  
INPUT STRING: a \* ( a + b )

E

T { + E }

P { \* T } { + E }

VAR { \* T } { + E }

a { \* T } { + E }

a \* T { \* T } { + E }

a \* ( E ) { \* T } { + E }

a \* ( T { + E } ) { \* T } { + E }

a \* ( P { \* T } { + E } ) { \* T } { + E }

a \* ( VAR { \* T } { + E } ) { \* T } { + E }

a \* ( a { \* T } { + E } ) { \* T } { + E }

a \* ( a { + E } ) { \* T } { + E }

a \* ( a + E ) { \* T } { + E }

a \* ( a + T { + E } ) { \* T } { + E }

a \* ( a + P { \* T } { + E }) { \* T } { + E }

a \* ( a + VAR { \* T } { + E }) { \* T } { + E }

a \* ( a + b { \* T } { + E }) { \* T } { + E }

a \* ( a + b { + E }) { \* T } { + E }

a \* ( a + b) { \* T } { + E }

a \* ( a + b) { + E }

a \* ( a + b)

**SECTION III**

**Answer each of the questions below using three sentences of fewer. (10)**

1. Provide a reason why “goto” might be considered harmful.  
     
   Code is difficult to read and maintain, is prone to errors
2. In the Swift programming language, why is the language able to not require parentheses around the test clause in an if statement?  
     
   Since the { is always required, the compiler always knows where the test clause ends
3. What does this code print out?

int var;

int B(int v) {

int var = v;

return var\*var; PRINTS OUT 5

}

int A() { return var \* 2; }

int main() {

int var = 5;

::var = B(var);

A();

cout << var << endl;

}

1. What is the benefit of overloading the operator [ ] in STL collections such as vector and map?  
     
   The code is more familiar/easier to read
2. Why must left-recursion be eliminated before building a recursive-descent parser?  
     
   Left recursion would cause a recursive descent parser to call itself forever and never terminate
3. Explain why Guardians of the Galaxy 2 will be the best movie of the summer. OR explain how you didn't know that there was a Guardians of the Galaxy 2 in the works.  
     
   OOOHGAH CHACKA OOOGHA OOGHA - OOOHGAH CHACKA OOOGHA OOGHA

SECTION IV

For each of the following pairs of regular expressions and strings, indicate if the regular expression matches the string. If it does not, indicate the reason that the string does not match the regular expression (20)

|  | **Regular Expression** | **String** | **Matches (Y for yes, N for no)** | **If mismatch, provide reason** |
| --- | --- | --- | --- | --- |
| 1. | [A-Z]?[a-z] | yikes | N | Pattern only allows one lowercase letter |
| 2. | ([A-Z][a-z]\*)+ | HelpMe | Y |  |
| 3. | [a-zA-Z0-9]\*\.cpp | prog..cpp | N | There are two dots in the string but only one in the pattern |
| 4. | [qwerty]+x | qqwyr | N | String does not end in an x |
| 5. | [Lsmft]?X+y | LXy | Y |  |
| 6. | \-?[1-9][0-9]+ | 275 | Y |  |
| 7. | [A-Z][a-z][A-Z] | AtoZ | N | String has 2 lowercase letters; pattern has only one |
| 8. | 0x[a-f0-9]+ | 0xBee5 | N | The pattern does not allow for the B |
| 9. | [a-z]\*[A-Z]+[a-v]\* | runAway | N | The letters after the A must be from the set a-v; the string contains a w (and a y) |
| 10. | [a-c]\-((de)|(fg))? | c-de | Y |  |